Amendment to the Claims

Listing of Claims:

Claims 1-39 . (cancelled)

Claim 40. (new) A computer system comprising:

an information storage subsystem comprising a contiguous linear portion which is logically divided into first and second heaps located at opposite ends of the storage subsystem, wherein any gap between the two heaps represents an unallocated region of storage, and wherein references are permitted from objects in the first heap to objects in the second heap and vice versa;

a garbage collector for operating across both heaps to remove objects that are no longer live;

a processor configured for expanding the first heap into the unallocated region according to a first expansion policy and for expanding the second heap into the unallocated region according to a second expansion policy; and

a transaction processing environment, wherein the first heap is used for storing objects that are deleted at the end of the current transaction, and the second heap is used for storing objects that persist from one transaction to another; and

wherein the first heap is reset to the same predetermined initial size at the start of each transaction, and the processor is structured to return an error condition when the second heap has expanded such that it is not possible to reset the first heap to its predetermined initial size.

Claim 41. (new) The computer system of claim 40, wherein a midpoint is defined halfway between the first heap and second heap, when each heap has its initial size.

Claim 42. (new) The computer system of claim 40, wherein the first expansion policy is always to expand into the unallocated region in order to satisfy a storage request.

Claim 43. (new) A computer system comprising:

an information storage subsystem comprising a contiguous linear portion which is logically divided into first and second heaps located at opposite ends of the storage portion subsystem, with any gap between the two heaps representing an unallocated region of storage, wherein references are permitted from objects on the first heap to objects on the second heap and vice versa;

a garbage collector for operating across both heaps to remove objects that are no longer live;

a processor configured for expanding the first heap into the unallocated region according to a first expansion policy and for expanding the second heap into the unallocated region according to a second expansion policy; and

a transaction processing environment, wherein the first heap is used for storing objects that are deleted at the end of the current transaction, and the second heap is used for storing objects that persist from one transaction to another; and wherein the first heap is reset to the same predetermined initial size at the start of each transaction, and a midpoint is defined halfway between the first heap and the second heap, when each heap has its initial size; the first expansion policy is always to expand into the unallocated region in order to satisfy a storage request; and the rate of expansion of the first heap into the unallocated region is slower once the first heap has passed the midpoint.

Claim 44 (new) . A computer system comprising:

an information storage subsystem comprising a contiguous linear portion which is logically divided into first and second heaps located at opposite ends of the storage portion subsystem, with any gap between the two heaps representing an unallocated region of storage, wherein references are permitted from objects on the first heap to objects on the second heap and vice versa;

a garbage collector for operating across both heaps to remove objects that are no longer live;

a processor configured for expanding the first heap into the unallocated region according to a first expansion policy and for expanding the second heap into the unallocated region according to a second expansion policy; and

a transaction processing environment, wherein the first heap is used for storing objects that are deleted at the end of the current transaction, and the second heap is used for storing objects that persist from one transaction to another; and

wherein the first heap is reset to the same predetermined initial size at the start of each transaction, the processor is structured to return an error condition if the second heap has expanded such that it is not possible to reset the first heap to its predetermined initial size, and the second expansion policy is to expand into the unallocated region in order to satisfy a storage request until the midpoint is reached, whereupon the system preferentially performs a garbage collection to satisfy the request.

Claim 45 (new) A computer system comprising:

an information storage subsystem comprising a contiguous linear portion which is logically divided into first and second heaps located at opposite ends of the storage portion subsystem, with any gap between the two heaps representing an unallocated region of storage, wherein references are permitted from objects on the first heap to objects on the second heap and vice versa;

a garbage collector for operating across both heaps to remove objects that are no longer live;

a processor configured for expanding the first heap into the unallocated region according to a first expansion policy and for expanding the second heap into the unallocated region according to a second expansion policy; and

a transaction processing environment, wherein the first heap is used for storing objects that are deleted at the end of the current transaction, and the second heap is used for storing objects that persist from one transaction to another; and

wherein the first heap is reset to the same predetermined initial size at the start of each transaction, and

the processor is structured to return an error condition if the second heap has expanded such that it is not possible to reset the first heap to its predetermined initial size, wherein the second expansion policy further includes trying to shrink the first heap to allow room to expand the second heap in order to satisfy a storage request.

Claim 46 (new) The computer system of claim 40, wherein the garbage collector performs a compact operation after a garbage collection of the first and second heaps, the compact operation being performed in response to a first set of criteria relevant to the first heap, and a second set of criteria relevant to the second heap.

Claim 47 (new) The computer system of claim 40, where the second set of criteria are more sensitive to fragmentation than the first set of criteria.

Claim 48 (new) The computer system of claim 40, further including means for shrinking the first and second heaps after compaction, and returning released storage to the unallocated region.

Claim 49 (new) The computer system of claim 40, further comprising:

a bit array, having one bit for each possible object location in the portion of storage, the bit indicating whether or not there is an object currently stored at the corresponding object location.

Claim 50 (new) A method of operating a computer system providing an object-based environment, the computer system including storage, a contiguous linear portion of which is logically divided into first and second heaps located at opposite ends of the storage portion, with any gap between the two heaps representing an unallocated region of storage, wherein references are permitted from objects on the first heap to objects on the second heap and vice versa, the method comprising steps of:

operating a garbage collector across both heaps to remove objects that are no longer live;

expanding the first heap into the unallocated region according to a first expansion policy;

expanding the second heap into the unallocated region according to a second expansion policy;

supporting a transaction processing environment wherein the first heap is used for storing objects that are deleted at the end of the current transaction, and the second heap is used for storing objects that persist from one transaction to another;

resetting the first heap to the same predetermined initial size at the start of each transaction; and

returning an error condition if the second heap has expanded such that it is not possible to reset the first heap to its predetermined initial size.

Claim 51 (new) The method of claim 50, further comprising a step of defining a midpoint halfway between the first heap and second heap, when each heap has its initial size.

Claim 52 (new) The method of claim 50, wherein the first expansion policy is always to expand into the unallocated region in order to satisfy a storage request.

Claim 53 (new) The method of claim 52, in which the rate of expansion of the first heap into the unallocated region is slower once the first heap has passed the midpoint.

Claim 54 (new) The method of claim 51, wherein the second expansion policy is to expand into the unallocated region in order to satisfy a storage request until the midpoint is reached, whereupon the system preferentially performs a garbage collection to satisfy the request.

Claim 55 (new) The method of claim 54, wherein the second expansion policy further includes trying to shrink the first heap to allow room to expand the second heap in order to satisfy a storage request.

Claim 56 (new) The method of claim 50, wherein the garbage collector performs a compact operation after a garbage collection of the first and second heaps, the compact operation being performed in response to a first set of criteria relevant to the first heap, and a second set of criteria relevant to the second heap.

Claim 57 (new) The method of claim 56, where the second set of criteria are more sensitive to fragmentation than the first set of criteria.

Claim 58 (new) The method of claim 56, further including steps of shrinking the first and second heaps after compaction, and returning released storage to the unallocated region.

Claim 59 (new) The method of claim 50, further including a step of providing a bit array, having one bit for each possible object location in the portion of storage, the bit indicating whether or not there is an object currently stored at the corresponding object location.

Claim 60 (new) A computer program product comprising program instructions recorded on a storage medium readable by a computer system, the computer system providing an object-based environment and including storage, a contiguous linear portion of which is logically divided into first and second heaps located at opposite ends of the storage portion, with any gap between the two heaps representing an unallocated region of storage, wherein references are permitted from objects in the first heap to objects in the second heap and vice versa, the program instructions causing the computer system to perform the steps of:

operating a garbage collector across both heaps to remove objects that are no longer live;

expanding the first heap into the unallocated region according to a first expansion policy;

for expanding the second heap into the unallocated region according to a second expansion policy;

wherein the computer system supports a transaction processing environment, and the first heap is used for storing objects that are deleted at the end of the current transaction, the second heap is used for storing objects that persist from one transaction to another, the first heap is reset to the same predetermined initial size at the start of each transaction, and the system returns an error condition if the second heap has expanded such that it is not possible to reset the first heap to its predetermined initial size.

Claim 61 (new) The computer program product of claim 60, wherein a midpoint is defined halfway between the first heap and second heap, when the heaps each have their initial size.

Claim 62 (new) The computer program product of claim 61, wherein the first expansion policy is always to expand into the unallocated region in order to satisfy a storage request.

Claim 63 (new) The computer program product of claim 62, in which the rate of expansion of the first heap into the unallocated region is slower once the first heap has passed the midpoint.

Claim 64 (new) The computer program product of claim 61, wherein the second expansion policy is to expand into the unallocated region in order to satisfy a storage request until the midpoint is reached, whereupon the system preferentially performs a garbage collection to satisfy the request.

Claim 65 (new) The computer program product of claim 64, wherein the second expansion policy further includes trying to shrink the first heap to allow room to expand the second heap in order to satisfy a storage request.

Claim 66 (new) The computer program product of claim 60, wherein the garbage collector performs a compact operation after a garbage collection of the first and second heaps, the compact operation being performed in response to a first set of criteria relevant to the first heap, and a second set of criteria relevant to the second heap.

Claim 67 (new) The computer program product of claim 66, wherein the second set of criteria are more sensitive to fragmentation than the first set of criteria.

Claim 68 (new) The computer program product of claim 66, wherein the program instructions further cause the computer system to perform steps of shrinking the first and second heaps after compaction, and returning released storage to the unallocated region.

Claim 69 (new) The computer program product of claim 60, wherein the program instructions further cause the computer system to perform the step of providing a bit array, having one bit for each possible object location in the portion of storage, the bit indicating whether or not there is an object currently stored at the corresponding object location.